UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/927,131	08/10/2001	Gowri Rajaram	UTL 00104	4248

32968 7590 01/04/2005

KYOCERA WIRELESS CORP.
P.O. BOX 928289
SAN DIEGO, CA 92192-8289

EXAMINER	
CHOW, CHIH CHING	
ART UNIT	**PAPER NUMBER**
2122	

DATE MAILED: 01/04/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/927,131	RAJARAM ET AL.	
	Examiner	Art Unit	
	Chih-Ching Chow	2122	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *10 August 2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-38* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-38* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *08/10/01* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *01/29/04*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the application filed on August 10, 2001.

2. The priority date considered for this application is August 10, 2001.

3. Claims 1-38 have been examined.

Drawings

4. New formal drawings in compliance with 37 CFR 1.121(d) are required in this application because the informal drawings is only sufficient for examination purpose. The corrected drawings are required in reply to the Office action to avoid abandonment of the application. The requirement for corrected drawings will not be held in abeyance.

Specification

5. The following information is missing from the specification:

Cross-References to Related Applications: See 37 CFR 1.78 and MPEP § 201.11.

Double Patenting

6. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970);and, *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

7. Claim 1 is provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claim 1 of copending Application No.09/969,305. Although the conflicting claims are not identical, they are not patentably distinct from each other, from the comparison listed in the following table:

Application (09,927,131) US 2003/0033599	***Co-Application (09,969,305) US 2003/0064717***
Claim 1	Claim 1
In a wireless communications device, a method for executing dynamic instruction sets, the method comprising:	In a wireless communications device, a method for managing system software download operations, the method comprising:
executing system software;	executing system software
launching a run-time engine;	launching a run-time engine
processing dynamic instruction sets;	processing dynamic instruction sets
operating on system data and system software; and,	
in response to operating on the system data and system software, controlling the execution of the system software (since it's a 'wireless communication device', it's inherited that the system receiving data via an airlink interface).	in response to processing the dynamic instruction sets, managing the downloading of system software updates received via an airlink interface.

Claim 1 of current application anticipates co-appliation claim 1 in that current claim 1 contains all the limitations of co-application claim 1. Specifically, "managing the downloading of system software updates" is one of the steps of "controlling the execution of the system software" as the dynamic instructions stes, which control the executionof the system software, permit the wireless device to "intelligently" or "conditionally" update the system software and system data. The downloading step is an inheret step of the software updating. Claim 1 of co-application therefore is not patentably distinct from the current application claim 1 and as such is unpatentable for obvous-type double patenting.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

8. Claim 3 (c) is provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claim 7 of copending Application No.09/927,131. Although the conflicting claims are not identical, they are not patentably distinct from each other, from the comparison listed in the following table:

Co-Application (09,927,131) US 2003/0033599	**Current Application (09,969,305) US 2003/0064717**
Claim 7	Claim 3
The method of claim 6 wherein receiving the dynamic instruction set includes receiving a **patch manager run time instruction (PMRTI) in a file system section (FSS) nonvolatile memory.**	(c) receiving **patch manager run tim instructions (PMRTIS) in a file system section (FSS) nonvolatile memory**, the patch manger run time instructions including dynamic instruction sets and new code sections.

Claim 7 of current application is anticipated by co-appliationclaim 3 (c) in that co-application claim 3(c) contains all the limitations of the current application claim 7. Claim 7 of the current application therefore is not patentably distinct from co-application claim 3(c) and as such is unpatentable for obvous-type double patenting.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

Claim Rejections - 35 USC § 103

9. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

10. Claims 1-5 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S Patent No. 6,493,871 by Thomas D. McGuire et al. (hereinafter "McGuire"), in view of U.S. Patent No. 6,023,620 by Lars Hansson (hereinafter "Hansson").

CLAIM	McGuire / Hansson
1. In a wireless communications device, a method for executing dynamic instruction sets, the method comprising: (a) executing system software; (b) launching a run-time engine; (c) processing dynamic instruction sets;	For item (a)-(c), see McGuire, column 5, lines 25-35, "the invention will be described in the general context of **computer-executable instructions** (*dynamic instruction sets*), such as program modules, being executed by a

(d) operating on system data and system software; and,

(e) in response to operating on the system data and system software, controlling the execution of the system software.

personal computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks (*dynamic instruction sets*) or implement particular abstract data types (*system data and system software*). Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multi-processor systems, microprocessor based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like." Also, McGuire column 8, lines 20-37, "After the SP5SETUP file 102 is **downloaded** to the client computer, it is executed to **self-extract the files** embedded therein. Of the **seven files** (*run-time engine*) extracted from SP5SETUP.EXE, the main files are the UPDATE.EXE file 106, which controls the remainder of the installation after the self-extractor runs, and the UPDATE.INF file 108, which is a script file that defines which files get copied, where they are copied to, etc. The SETUPAPI.DLL file 110 contains general-purpose file installation subroutines that are used by UPDATE.EXE. The SPMSG.DLL file 112 contains all the localized dialogs and messages needed by UPDATE.EXE for multi-language support. The EULA.TXT file 114 and the README.TXT file 116

are the end-user license agreement and "read me" files, which UPDATE.EXE will display for the user's consent before installation. The SPUNINST.EXE file 118 is the un-install utility, supplied in case the user wishes later to remove the installed updates. The operations of UPDATE.EXE and how the elements of the script file UPDATE.INF are described in greater detail below." For item d, see McGuire claim 50, column 20, lines 32-34, "**processing the download d files** to update the existing files... (*managing the downloading of system software updates*)". McGuire teaches all aspects of claim 1, but he does not mention 'wireless communication', and 'airlink' specifically, however, Hansson teaches it in an analogous prior art. In Hansson, column 1, lines 42-43, "The present invention comprises a method and apparatus for downloading software into a remotely located cellular telephone via wireless communication." It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement McGuire's disclosure of the software download system by using wireless communication interface taught by Hansson, for the purpose of reprogram a cellular telephone remotely (Hansson, column 1, lines 35-36).

2. The method of claim 1 further comprising:

For the feature of claim 1 see claim 1 rejection. For the rest of the feature in

following the processing of the dynamic instruction sets, deleting dynamic instruction sets.

claim 2, see McGuire column 14, lines 56-64, "In the final step of the downloading process, the setup logic is run again, but this time file copies and other system changes will be allowed to occur. Whenever UPDATE.EXE needs to copy a 'new' file, it will obtain that file from the temporary directory where the **received** files were reconstituted (*dynamic instruction sets*). Because the complete installation file set is now present on the client, the setup program will be able to run to completion and properly upgrade the system. After all the files have been copied into their proper directories, the files in the temporary directory are **deleted**, then the files extracted from initial setup package SP5SETUP.EXE are also deleted."

3. The method of claim 1 wherein processing dynamic instruction sets includes processing instructions in response to mathematical and logical operations.

For the feature of claim 1 see claim 1 rejection. Any instruction is possible, including mathematical and logical operations.

4. The method of claim 3 further comprising:
receiving the dynamic instruction sets.

For the feature of claim 3 see claim 3 rejection. See claim rejection 1, McGuire's disclosure teaches receiving dynamic instruction sets.

5. The method of claim 4 wherein receiving the dynamic instruction sets includes receiving the dynamic instruction sets through an interface

For the feature of claim 4 see claim 4 rejection. See claim rejection 1, Hansson's disclosure teaches receiving data via airlink, RF, ... (*wireless*

selected from the group including airlink, radio frequency (RF) hardline, installable memory module, infrared, and logic port interfaces.

communication) etc.

11. Claims 20-23, 38 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S Patent No. 6,493,871 by Thomas D. McGuire et al. (hereinafter "McGuire"), in view of U.S. Patent No. 6,023,620 by Lars Hansson (hereinafter "Hansson"), and further in view of U.S. Patent No. 5,930,704 by David L. Kay (hereinafter "Kay").

CLAIM	**McGuire / Hansson / Kay**
20. In a wireless communications device, a dynamic instruction set execution system, the system comprising: (a) executable system software and system data differentiated into code sections; (b) dynamic instruction sets for operating on the system data and the system software, and controlling the execution of the system software; and, (c) a run-time engine for processing the dynamic instruction sets.	McGuire and Hansson teach all aspects of claim 20, but he does not mention 'Code Sections' specifically, however, Kay teaches it in an analogous prior art. In Kay, Kay column 14, lines 52-54, "the code held in each of the flash memories is structured into a boot-strap and loader code and a **separate main cod section**." And column 16, lines 43-49, As mentioned above with reference to FIG. 19, the code in the flash memories 310 and 312 is split into **two distinct sections**, the boot-strap and loader section and the main code section. ..The boot strap and loader **code section** forms an independent executable segment which resides in the boot areas 610/612 of the flash memories". It would have been obvious to a person

	of ordinary skill in the art at the time of the invention was made to supplement McGuire's disclosure of the software download system by using code section taught by Kay, for the purpose of sustaining a small section of the code to ensure reliability in the event of a down-load failure. (Kay column 14, lines 41-44).
21. The system of claim 20 wherein the run-time engine processes dynamic instruction sets to perform mathematical and logical operations.	For the feature of claim 20 see claim 20 rejection. Any instruction is possible, including mathematical and logical operations.
22. The system of claim 21 further comprising: a file system section nonvolatile memory for receiving the dynamic instruction sets.	For the feature of claim 21 see claim 21 rejection. See McGuire's FIG. 1, it teaches a system with nonvolatile memory for receiving the dynamic instruction sets.
23. The system of claim 22 further comprising: an interface through which the dynamic instruction sets are received into the file system section, wherein the interface is selected from the group including airlink, radio frequency (RF) hardline, installable memory module, infrared, and logic port interfaces.	For the feature of claim 22 see claim 22 rejection. For the rest of the claim 23 feature see claim 5 rejection.
38. In a wireless communications device, a dynamic instruction set execution system, the system comprising: executable system software and system data differentiated into code sections with symbol libraries arranged	Claim 38 is the same as claim 1 rejection except the PMRTIS part. For the PMRTI feature see McGuire, claim 5, "processing includes determining whether the **download d fil s includ s a patch** for said first existing file, and

within;

patch manager run time instructions (PMRTIS) organized as dynamic instruction sets with operation code and data items for operating on the system data and the system software, and for controlling the execution of the system software;

a file system section nonvolatile memory for receiving the patch manager run time instructions; and,

a run-time library arranged in a first code section for processing the dynamic instruction sets.

updating said first existing file with the **patch**", also in Claim 40, "whether a patch or a full file corresponding to said each requested file is requested; when a full file is requested, including a full file corresponding to said each requested file in a download reply; when a patch is requested, determining whether said patch is in a download database of the download server, and (i) when said **patch** is in the download database, including said patch in the download reply; (ii) when said patch is not in the download database, including a full file corresponding to said each requested file in the download reply;(c) transmitting the download reply to the client computer." During the runtime, the patch code will be executed.

12. Claims 6-9, 19, 24-28 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S Patent No. 6,493,871 by Thomas D. McGuire et al. (hereinafter "McGuire"), in view of U.S. Patent No. 6,023,620 by Lars Hansson (hereinafter "Hansson"), further in view of U.S. Patent No. 5,930,704 by David L. Kay (hereinafter "Kay"), further in view of US 2002/0019973 by Seiji Hayashida (hereinafter "Hayashida").

CLAIM	McGuir / Hansson / Kay / Hayashida
6. The method of claim 5 further comprising: (a) forming the system software into symbol libraries, each symbol library comprising symbols having related functionality; (b) arranging the symbol libraries into code sections; and, wherein launching a run-time engine includes invoking a run-time library from a first code section.	For the feature of claim 5 see claim 5 rejection. For item (b) see claim 20 rejection. McGuire, Hansson and Kay teach all aspects of claim 5, but he does not mention 'symbol libraries' specifically, however, Hayashida teaches it in an analogous prior art. In Hayashida, paragraph 74, "the details of the intrinsics function definition are recorded in a **symbol table** (step S35), which is a table used for searching for defined intrinsics functions and their arguments." And paragraph 75, "A check is performed to determine whether or not the specified identifier (for example, mov) is **stored in the symbol table as an intrinsics function** in the intrinsics information database 18 (step S43)." It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement McGuire's disclosure of the software download system by using symbol table taught by Hayashida, for the purpose of searching for a defined function easier (Hayashida paragraph 74).
7. The method of claim 6 wherein receiving the dynamic instruction set includes receiving a patch manager run time instruction (PMRTI) in a file system section nonvolatile memory.	For the feature of claim 6 see claim 6 rejection. For the PMRTI feature see McGuire, claim 5, "processing includes determining whether the **downloaded files includes a patch** for said first existing file, and updating said first existing file with the **patch**", also in

	Claim 40, "whether a patch or a full file corresponding to said each requested file is requested; when a full file is requested, including a full file corresponding to said each requested file in a download reply; when a patch is requested, determining whether said patch is in a download database of the download server, and (i) when said **patch** is in the download database, including said patch in the download reply; (ii) when said patch is not in the download database, including a full file corresponding to said each requested file in the download reply;(c) transmitting the download reply to the client computer." During the runtime, the patch code will be executed.
8. The method of claim 7 wherein receiving the patch manager run time instructions includes receiving conditional operation code and data items; wherein processing dynamic instruction sets includes: (a) using the run-time engine to read the patch manager run time instruction operation code; and, (b) performing a sequence of operations in response to the operation code.	For the feature of claim 7 see claim 7 rejection. For the rest of the claim 8 feature see claim 1 rejection.
9. The method of claim 8 wherein processing dynamic instruction sets includes:	For the feature of claim 8 see claim 8 rejection. For the rest of the claim 9 feature see claim 1 rejection.

(a) using the run-time engine to capture the length of the patch manager run time instruction;
(b) extracting the data items from the patch manager run time instruction, in response to the operation code; and,
(c) using the extracted data in performing the sequence of operations responsive to the operation code.

19. In a wireless communications device, a method for executing dynamic instruction sets, the method comprising:
(a) forming the system software into symbol libraries, each symbol library comprising symbols having related functionality;
(b) arranging the symbol libraries into code sections in a code storage section nonvolatile memory;
(c) executing system software;
(d) receiving a patch manager run time instruction (PMRTI), including conditional operation code and data items, in a file system section nonvolatile memory;
(e) calling a run-time library from a first code section;
(f) processing the patch manager run time instruction operation code;
(g) operating on system data and system software; and,
(h) in response to operating on the system data and system software, controlling the execution of the system software.

See claim 1, 20, and 6 rejections.

24. The system of claim 23 wherein the executable system software and system data include symbol libraries, each symbol library comprising symbols having related functionality, arranged into code sections; and, wherein the run-time engine is a run-time library arranged in a first code section.	For the feature of claim 23 see claim 23 rejection. For the rest of the claim 24 feature see claim 6 and 20 rejections.
25. The system of claim 24 wherein the dynamic instruction sets include conditional operation code and data items, and wherein the dynamic instruction sets are organized in a patch manager run time instruction (PMRTI).	For the feature of claim 24 see claim 24 rejection. For the rest of the claim 25 feature see claim 1 and 7 rejections.
26. The system of claim 25 further comprising: a code storage section nonvolatile memory for storing code sections.	For the feature of claim 25 see claim 25 rejection. For the rest of the claim 26 feature see claim 1 rejection (see McGuire FIG. 1).
27. The system of claim 26 wherein the run-time engine reads the dynamic instruction set operation code and performs a sequence of operations in response to the operation code.	For the feature of claim 26 see claim 26 rejection. For the rest of the claim 27 feature see claim 1 rejection.
28. The system of claim 27 wherein the run-time engine captures the length of a dynamic instruction set to determine if data items are included, extracts the data items from the dynamic instruction set, and uses the extracted data in performing the sequence of operations responsive to the operation code.	For the feature of claim 27 see claim 27 rejection. For the rest of the claim 28 feature see claim 1 rejection.

13. Claims 10-18, 29-37 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S Patent No. 6,493,871 by Thomas D. McGuire et al. (hereinafter "McGuire"), in view of U.S. Patent No. 6,023,620 by Lars Hansson (hereinafter "Hansson"), further in view of U.S. Patent No. 5,930,704 by David L. Kay (hereinafter "Kay"), and further in view of US 2002/0019973 by Seiji Hayashida (hereinafter "Hayashida"), and further in view of US Patent No. 6,442,660 by Paul Henerlau et al. (hereinafter "Henerlau").

CLAIM	**McGuire / Hansson / Kay Hayashida / Henerlau**
10. The method of claim 9 wherein arranging the symbol libraries into code sections includes starting symbol libraries at the start of code sections and arranging symbols to be offset from their respective code section start addresses; the method further comprising: (a) storing the start of code sections at corresponding start addresses; (b) maintaining a code section address table cross-referencing code section identifiers with corresponding start addresses; and, (c) maintaining a symbol offset address table cross-referencing symbol identifiers with corresponding offset addresses, and corresponding code section identifiers.	For the feature of claim 9 see claim 9 rejection. McGuire, Hansson, Kay and Hayashida teach all aspects of claim 10, but he does not mention 'code section address table cross-referencing' specifically, however, Henerlau teaches it in an analogous prior art. In Henerlau column 8, lines 24-30, "in creating the relocation table data is to take the relocation file and compress the data. It should be noted that for a given relocation value, the **addresses** associated with the value are often close to one another, usually within 256 bytes, and it is therefore possible to create a smaller table, by just **storing the start address and the offset** *(code sections),* or delta, to the next address. The **start addr ss** requires a full four bytes, 32 bits, while the offset

	quantities require only one byte." It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement McGuire, Hansson, Kay and Hayashida's disclosure of the software download system by using code section taught by Henerlau, for the purpose of providing knowledge for various versions of software when updating software (Henerlau, column 2, lines 16-26).
11. The method of claim 10 wherein receiving the patch manager run time instruction includes receiving symbol identifiers; the method further comprising: (a) locating symbols corresponding to the received symbol identifiers by using the code section address table and symbol offset address table; wherein performing a sequence of operations in response to the operation code includes: when the located symbols are data items, extracting the data; and, when the located symbols are instructions, executing the symbols.	For the feature of claim 10 see claim 10 rejection. For the symbol identifiers see claim 6 rejection.
12. The method of claim 8 wherein processing dynamic instruction sets includes: (a) accessing system data stored in a second code section in the file system section; (b) analyzing the system data;	For the feature of claim 8 see claim 8 rejection. For the rest of claim 12. See Henerlau, column 2, lines 1-10, "method of dynamic system relocation, including creating a ROM version of an embedded application which is executable from ROM; creating a RAM version of the

(c) creating updated system data;
(d) wherein operating on system data and system software includes replacing the system data in the second section with the updated system data; and,
(e) wherein controlling the execution of the system software includes using the updated system data in the execution of the system software.

embedded application which is executable from RAM; **comparing** the RAM version of the embedded application to the ROM version of the embedded application to identify differences between the RAM version and the ROM version; **storing the differences between the ROM version and the RAM version in a relocation table"**.

13. The method of claim 8 further comprising:
storing a plurality of code sections in a code storage section nonvolatile memory;
wherein processing dynamic instruction sets includes:
(a) accessing system data stored in a third code section in the code storage section;
(b) analyzing the system data; creating updated system data;
(c) wherein operating on the system data and system software includes replacing the system data in the third code section with the updated system data; and,
(d) wherein controlling the execution of the system software includes using the updated system data in the execution of the system software.

For the feature of claim 8 see claim 8 rejection. For the rest of claim 13 features see claim 6, 7 and 12 rejections.

14. The method of claim 8 further comprising:
storing a plurality of code sections in a code storage section nonvolatile

For the feature of claim 8 see claim 8 rejection. For the rest of claim 14 features see claim 10and 12 rejections.

memory;
loading read-write data into volatile memory;
wherein processing dynamic instruction sets includes:
(a) accessing the read-write data in volatile memory;
(b) analyzing the read-write data; creating updated read-write data;
(c) wherein operating on the system data and system software includes replacing the read-write data in volatile memory with the updated read-write data; and,
(d) wherein controlling the execution of the system software includes using the updated read-write data in the execution of the system software.

15. The method of claim 8 wherein processing dynamic instruction sets includes:
in response to the operation code, monitoring the execution of the system software;
collecting performance data;
storing the performance data; and,
wherein operating on the system data and system software includes using the performance data in the evaluation of system software.

For the feature of claim 8 see claim 8 rejection. For the rest of the claim 15 feature see claim 12 rejection.

16. The method of claim 15 further comprising:
transmitting the stored data via an airlink interface.

For the feature of claim 15 see claim 15 rejection. Airlink feature is disclosed in Hansson's art (see claim 1 rejection).

17. The method of claim 8 further comprising: (a) storing a plurality of code sections in a code storage section nonvolatile memory; (b) wherein receiving patch manager run time instructions includes receiving a new code section; (c) wherein operating on the system data and system software includes adding the new code section to the code storage section; and, (d) wherein controlling the execution of the system software includes using the new code section in the execution of the system software.	For the feature of claim 8 see claim 8 rejection. See claim 1 rejection, in McGuire's disclosure teaches using new code in the execution of the system software.
18. The method of claim 17 wherein receiving a new code section includes receiving an updated code section; and, wherein operating on the system data and system software includes replacing a fourth code section in the code storage section with the updated code section.	For the feature of claim 17 see claim 17 rejection. In Henerlau disclosure, it can be plurality of code sections. See claim 10 rejection.
29. The system of claim 28 wherein the symbol libraries are arranged to start at the start of code sections and symbols are arranged to be offset from their respective code section start addresses; wherein a code storage section includes start addresses corresponding to code section start addresses;	For the feature of claim 28 see claim 28 rejection. For the rest of the claim 29 feature see claim 1, 6 and 10 rejections.

the system further comprising:

a code section address table cross-referencing code section identifiers with corresponding start addresses in the code storage section; and,

a symbol offset address table cross-referencing symbol identifiers with corresponding offset addresses, and corresponding code section identifiers.

30. The system of claim 27 wherein the dynamic instruction set includes symbol identifiers; and,

wherein the run-time engine locates symbols corresponding to the received symbol identifiers using the code section address table and symbol offset address table, extracts data when the located symbols are data items, and executes the symbols when the located symbols are instructions.

For the feature of claim 27 see claim 27 rejection. For the rest of the claim 30 feature see claim 1, 6 and 10 rejections.

31. The system of claim 27 wherein the system data is stored in a second code section in the file system section;

wherein the run-time engine accesses system data, analyzes the system data, creates updated system data, and replaces the system data in the second code section with the updated system data in response to the operation code; and,

wherein the system software is controlled to execute using the updated system data.

For the feature of claim 27 see claim 27 rejection. For the rest of the claim 31 feature see claim 1, 6, 10, and 18 rejection.

32. The system of claim 27 wherein the system data is stored in a third code section in the code storage section;

wherein the run-time engine accesses system data, analyzes the system data, creates updated system data, and replaces the system data in the third code section with the updated system data in response to the operation code; and,

wherein the system software is controlled to execute using the updated system data.

For the feature of claim 27 see claim 27 rejection. For the rest of the claim 32 feature see claim 1, 6, 10 and 18 rejections.

33. The system of claim 27 further comprising:

a volatile memory to accept read-write data;

wherein the run-time engine accesses the read-write data, analyzes the read-write data, creates updated read-write data, and replaces the read-write data in the volatile memory with the updated read-write data in response to the operation code; and,

wherein the system software is controlled to execute using the updated read-write data in volatile memory.

For the feature of claim 27 see claim 27 rejection. For the rest of the claim 33 feature see claim 1, 6, and 10 rejections.

34. The system of claim 27 wherein the run-time engine monitors the execution of the system software, collects performance data, and stores the performance data in the file system section in response to the operation code; and,

For the feature of claim 27 see claim 27 rejection. For the rest of the claim 34 feature see claim 1, 6, and 10 rejections.

wherein the system software is controlled to execute by collecting the performance data for evaluation of the system software.

35. The system of claim 34 wherein the run-time engine accesses the performance data from the file system section and transmits the performance data via an airlink interface in response to the operation code.

For the feature of claim 34 see claim 34 rejection. For the rest of the claim 35 feature see claim 1 and 6 rejections.

36. The system of claim 27 wherein the file system section receives a patch manager run time instruction including a new code section;

wherein the run-time engine adds the new code section to the code storage section in response to the operation code; and,

wherein the system software is controlled to execute using the new code section.

For the feature of claim 27 see claim 27 rejection. For the rest of the claim 36 feature see claim 1, 6 and 10 rejections.

37. The system of claim 36 wherein the file system section receives a patch manager run time instruction including an updated code section;

wherein the run-time engine replaces a fourth code section in the code storage section with the updated code section in response to the operation code; and,

wherein the system software is controlled to execute using the updated code section.

For the feature of claim 36 see claim 36 rejection. For the rest of the claim 36 feature see claim 1, 6, 10 and 18 rejections.

Conclusion

14. 35 USC § 103 rejection: Claims 1-38.

15. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Do, U.S. Patent No. 6,415,266 discloses a method for using dynamic instruction system based on vehicle type information performs computations on a variety of data in a production line.

Lillich, U.S. Patent No. 5,790,856 discloses a method for forming the system software into a first plurality of symbol libraries, each symbol library comprising at least one symbol.

Beasley, U.S. Patent No. 5,699,275 discloses a method for updating system software stored in memory comprises a patch library (patch bank).

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 571-272-3693. The examiner can normally be reached on 7:00am - 3:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Chih-Ching Chow
Examiner
Art Unit 2122

ANTONY NGUYEN-BA
PRIMARY EXAMINER

CC